Inverse Random Under Sampling (IRUS) on Class Imbalance, Heart Stroke Dataset

Jaffar bin Farooq (18K-1294), Muhammad Hasan (18K-0294), Muhammad Irham Rahim(18K-0150)

Data Science Project

1. Introduction

Imbalanced classifications pose a challenge for predictive modeling as most of the machine learning algorithms used for classification were designed around the assumption of an equal number of examples for each class. This results in models that have poor predictive performance, specifically for the minority class. This is a problem because typically, the minority class is more important and therefore the problem is more sensitive to classification errors for the minority class than the majority class. In this project, we aimed to work on the Heart Stroke dataset, with considerable class imbalance. We implemented Inverse Random Under Sampling method and compared the results of it with the raw data results.

2. Workflow

2.1 Research Goal

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient. Our research goal is to predict stroke occurrence from the class imbalanced dataset using IRUS model and find the correct accuracy using this model and afterwards we have to compare the model's performance before and after applying the IRUS algorithm on the said given class imbalanced dataset.

2.2 Retrieving Data

The dataset which is used for this project is to predict whether a patient is likely to get stroke or not. It is an open-source dataset available on Kaggle. https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset. The dataset contains 12 attributes in which 11 attributes are features and 1 is class label. The attributes of the datasets are:

1)id: unique identifier
2) gender: "Male", "Female" or "Other"
3) age: age of the patient
4) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
5) heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
6) ever_married: "No" or "Yes"
7) work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
8) Residence_type: "Rural" or "Urban"
9) avg_glucose_level: average glucose level in blood
10) bmi: body mass index
11) smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*
12) stroke: 1 if the patient had a stroke or 0

if not
*Note: "Unknown" in smoking_status means that the information is unavailable for this patient

The dataset contains 5110 patient records, 249 of them are had stroke and the rest 4861 records didn't had stroke. And that's why the stroke class is imbalance.

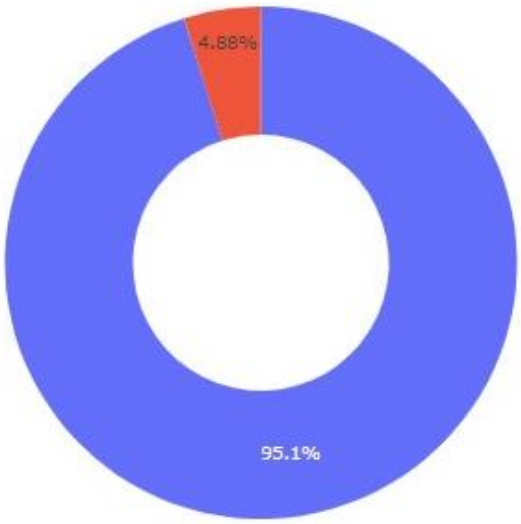


Figure: Class Distribution

2.3 Data Preparation

In data pre-processing first we have checked if there are any null values or not

```
df.isnull().sum()
```

```
id                   0
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                201
smoking_status       0
stroke               0
```

There were 201 null values in BMI column we have filled them with BMI mean value

```
# Fill null values in bmi with mean value
df['bmi'].fillna(df['bmi'].mean(),inplace=True)
```

We have dropped the id column because that does not matter.

```
df.drop(['id'],axis=1,inplace=True)
```

We have removed smoking_status with values 'unknown' and then replaced the missing values using KNNImputer.

```
from sklearn.impute import KNNImputer
imputer=KNNImputer(n_neighbors=3)
df_filled=imputer.fit_transform(df)
df_filled
```

```
df = pd.DataFrame(df_filled,columns=df.columns)
df.head()
```

We have performed Feature selection based on correlation

```
plt.figure(figsize=(20,15))
sns.heatmap(df.corr())
plt.show()
```

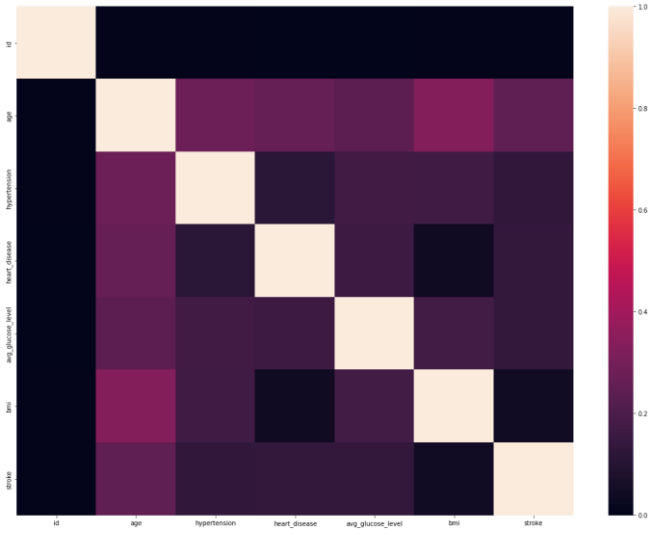


Figure: Correlation heatmap

```
print(df.corr()['stroke'].sort_values())
```

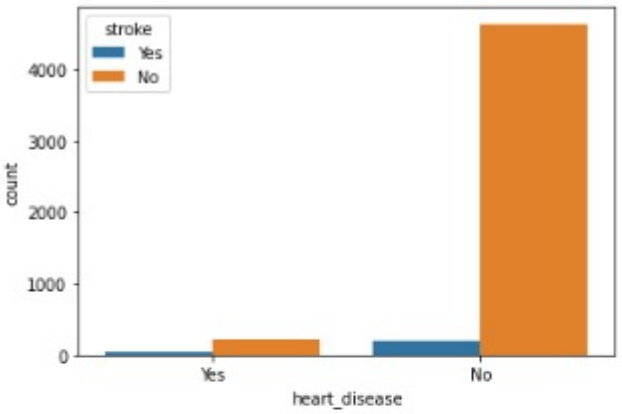
```
id                   0.006388
bmi                  0.042374
hypertension         0.127904
avg_glucose_level    0.131945
heart_disease        0.134914
age                  0.245257
stroke               1.000000
Name: stroke, dtype: float64
```

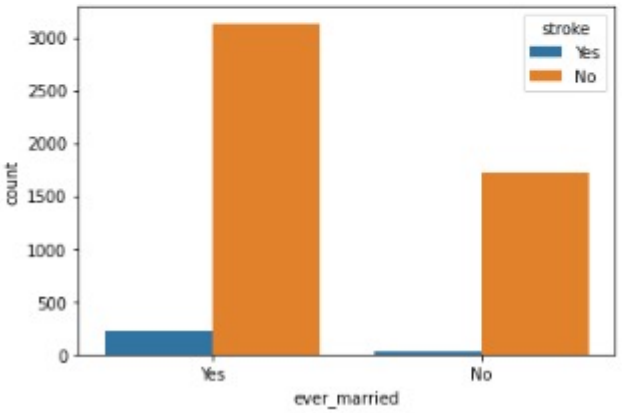
```
#drop columns with -0.02<correlation<0.02¶
cols_to_drop=['Residence_type','gender']
df = df.drop(cols_to_drop, axis = 1).copy()
df.head(10).T
```

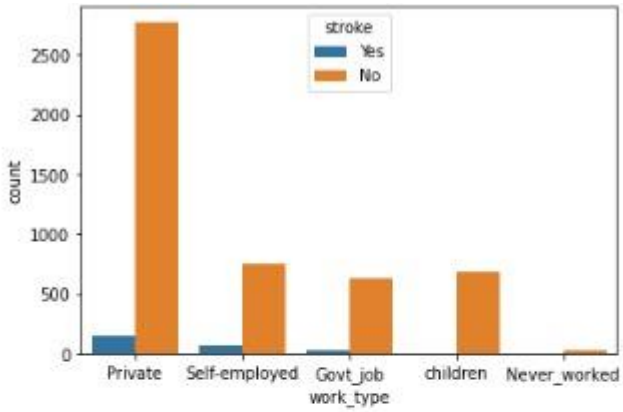
2.4 Data Exploration

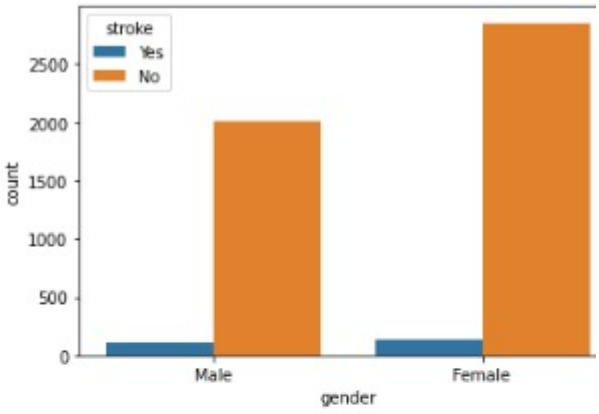
After cleaning the data, we have gathered the information about the data and how every attribute is distributed quantitively.

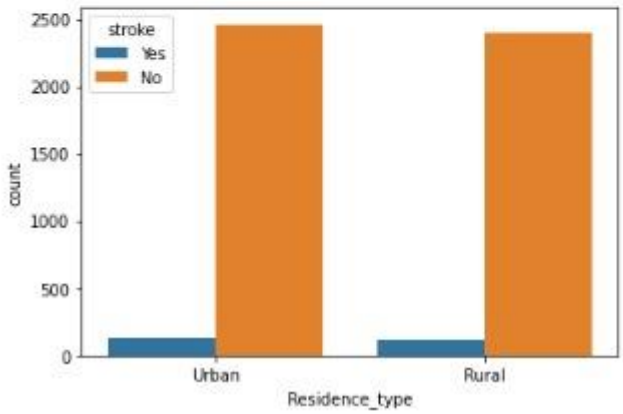
We have used count plot to find number of strokes in each discrete feature.

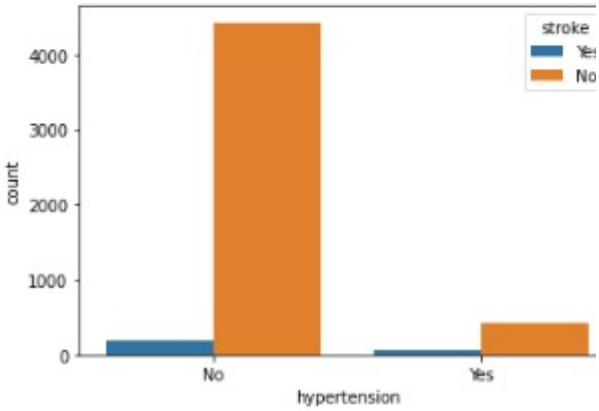


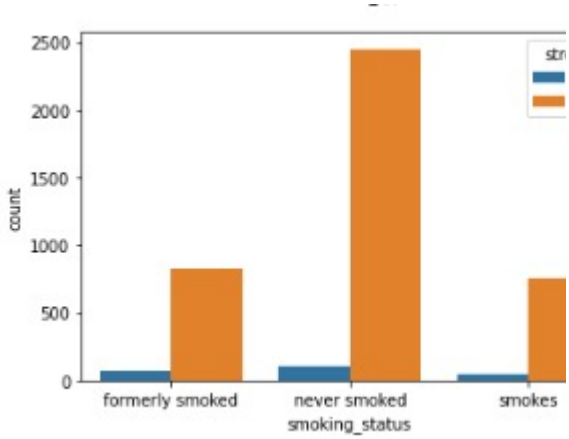


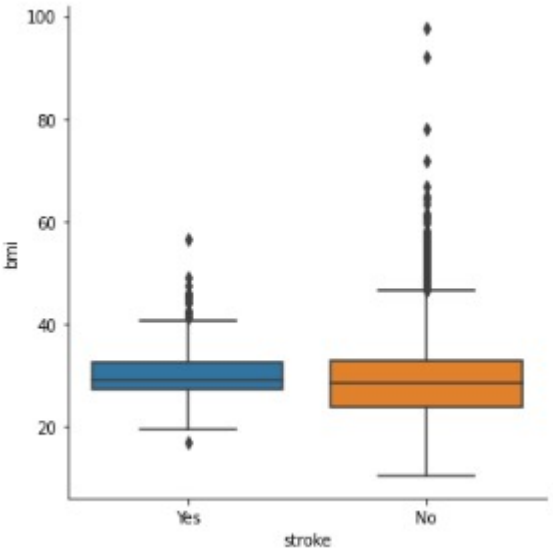




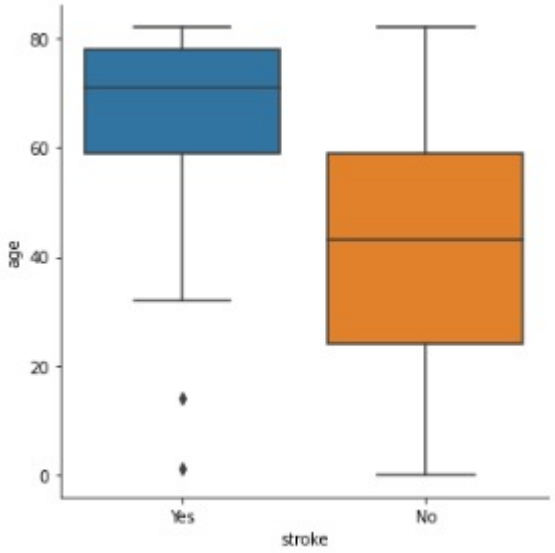


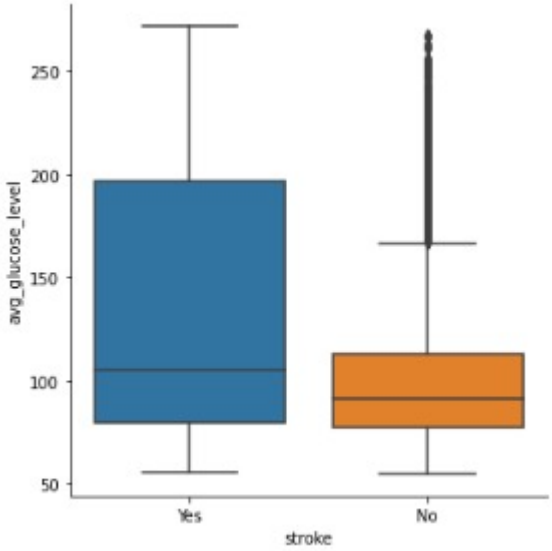






We have used box and whisker plots to find number of strokes in each continuous feature.





By visualizing the graphs, we have assumed the following:

1. Being unmarried reduces your risk of a stroke
2. Being a smoker or a formerly smoker increases your risk of having a stroke
3. more than 25% of stroke cases They had hypertension
4. Female and male both have equal number of stroke cases while there is not any single case of stroke in other gender type.
5. Patient with private job have more number stroke cases then patient who are self employed or have a government job
6. Stroke has the highest correlation with age.
7. Patients with stroke having higher avg_glucose_level.

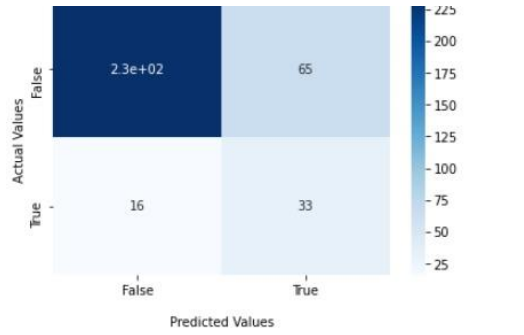
2.5 Data Modelling

In this section we compared the performance of 4 models with each other with and without holdout approach. The train and test sets were divided using Hold-Out approach in an 80:20 ratio using the following models:

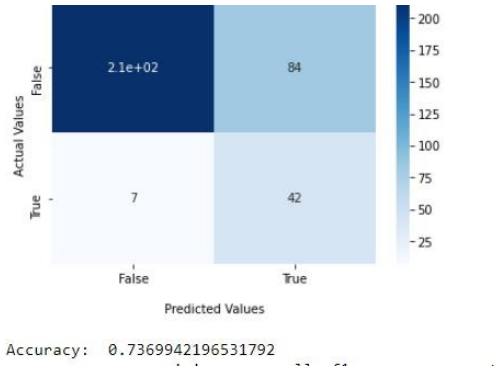
- Decision Tree
  Hyper Parameters= criterion='gini', splitter='best'
- Random Forest
- Neural Networks
  Hyper Parameters= hidden_layer_sizes = (150,100,50), max_iter=300,activation = 'relu',solver='adam',random_state= 1
- Logistic Regression
  Hyper Parameters= solver='liblinear', random_state=0

Confusion Matrix and Accuracies with IRUS:

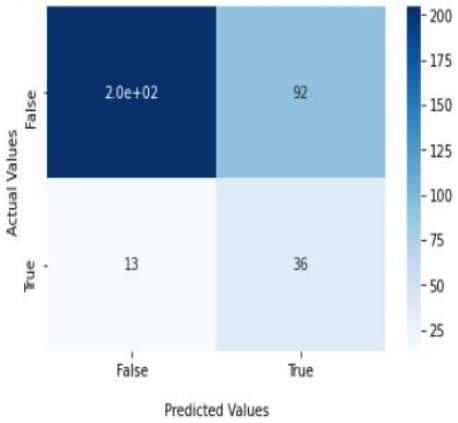
Decision tree



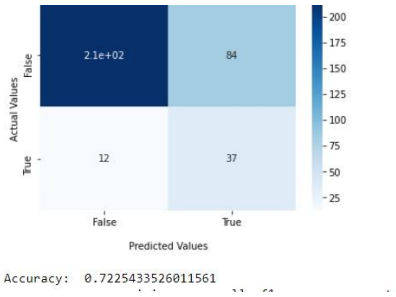
Random forest



Neural Networks

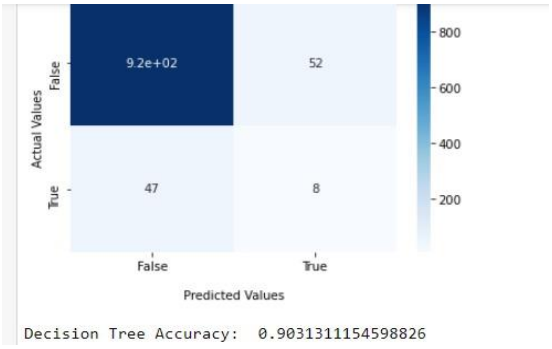


Logistic Regression

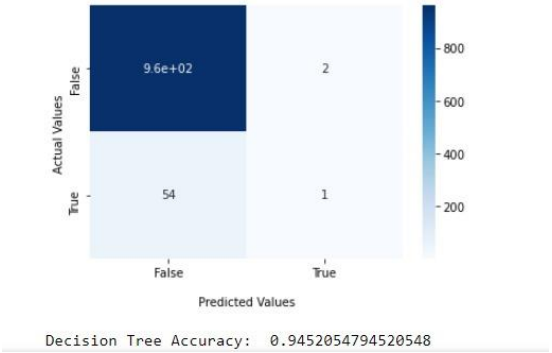


Confusion Matrix and Accuracies without IRUS:

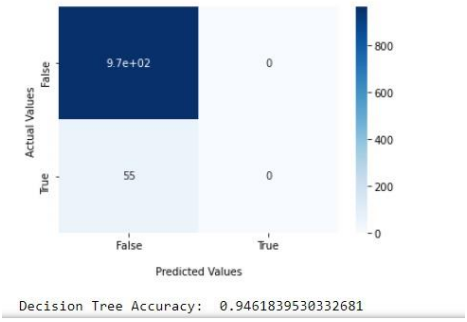
Decision tree



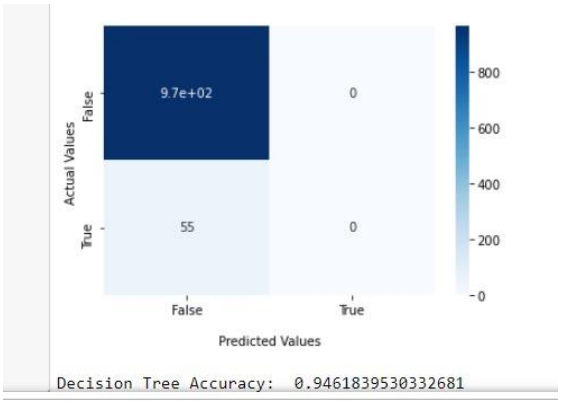
Random forest



Neural Network



Logistic regression



3. Conclusion

After applying IRUS the minority class of our dataset (stroke: 1) is more correctly classified than before and therefore we have catered the problems caused by this class imbalanced dataset.